Outline and Tutorial for the OpenBayes Python Module.

Johannes Castner

December 19, 2013

Contents

1 Inheritance and Module Organitation

This is a rough outline of what is contained within the OpenBayes Python module:

1.1 The *init* file

This file simply imports all of the relevant moduls accessed by users:

```
__all__ = ['bayesnet', 'distributions', 'inference',
'potentials', 'table', 'graph', 'OpenBayesXBN', 'BNController']
```

1.2 The *bayesnet* file

This file contains three classes: `'BVertex'`, `'BNet'` and `'BNetTestCase'` but it only exports two of them to the end user:

```
__all__ = ['BVertex', 'BNet']
```

1.2.1 BVertex:

This class inherits from `graph.Vertex` and represents one random variable. It is initiated with a name space dictionary containing: `name`, the name of the variable, `discrete` an indicator of whether or not the variable is discrete or continuous, which is either true, meaning discrete (the default) or false, meaning continuous, `nvalues` the number of possible values that the random variable can take on (if it is indeed descrete) and `observed`, which is true if the variable can be observed and false if it is a hidden variable.

```
{'name': 'b', 'family': [<bayesnet.BVertex object at 0x2934a10>],
'observed': True, 'discrete': True, 'nvalues': 2, 'distribution': None, '_e': []}
```

1.2.2 The methods of BVertex:

The methhods of `BVertex` all refer to the `distributions` file, desrcibed further below.

- `InitDistribution(self, *args, **kwargs)`:

initializes conditional distribution of the variable as a function of incoming (causal nodes).

- `setDistributionParameters(self, *args, **kwargs)`

does what its name suggests: it sets the parameters of the distribution.

- `GetSamplingDistribution(self)`

  used for the MCMC engine to sample from the distribution.

- `__cmp__(a,b)`

  This method is used for correct message passing.

1.2.3 BNet:

This class inherits from `graph.Graph` and it uses the `logging.getLogger('BNet')` class from the `logging` sub-module. It initializes exactly as `graph.Graph.__init__(self, name)`. Its name-space dictionary looks as:

```
{'e': {}, 'name': 'b', 'v': {}}
```